Chapter

Middleware Solutions for the Internet of Things: A Survey

Mehdia Ajana El Khaddar

Abstract

The Internet of Things (IoT), along with its wider variants including numerous technologies, things, and people: the Internet of Everything (IoE) and the Internet of Nano Things (IoNT), are considered as part of the Internet of the future and ubiquitous computing allowing the communication among billions of smart devices and objects, and have recently drawn a very significant research attention. In these approaches, there are varieties of heterogeneous devices empowered by new capabilities and interacting with each other to achieve specific applications in different domains. A middleware layer is therefore required to abstract the physical layer details of the smart IoT devices and ease the complex and challenging task of developing multiple backend applications. In this chapter, an overview of IoT technologies, architecture, and main applications is given first and then followed by a comprehensive survey on the most recently used and proposed middleware solutions designed for IoT networks. In addition, open issues in IoT middleware design and future works in the field of middleware development are highlighted.

Keywords: Internet of Things (IoT), WSNs, radio frequency identification (RFID), virtual machine, events, services, middleware architecture, context awareness, ubiquitous computing, machine-to-machine (M2M) communication

1. Introduction

Nowadays, various new generation-connected objects or things are invading our daily lives including sensors, radio frequency identification (RFID) tags, smartphones, wearables, and actuators among others, due to the emergence of new technologies. With the development of cloud computing and wireless technologies, and the emergence of new connected devices at a decreasing price, the IoT market is expected to grow rapidly fostering the development of applications in different domains, including but not limited to healthcare, manufacturing, logistics and transportation, traffic management, home automation, smart cities, smart grids, smart agriculture, etc. [1]. These applications will use the raw data generated by the different connected things/objects and provide new services in the targeted domains [2]. The Global System for Mobile Communications Association (GSMA) forecasts that "by 2025, the IoT connections will reach almost 25 billion globally" [3]. These predictions are therefore highlighting the role of IoT in providing new ways of communication over the Internet.

The IoT network is considered a heterogeneous network with a complex structure, connecting a wide range of devices using different evolving technologies such as Bluetooth, ZigBee, Wi-Fi, 3G, 4G, 5G. The ubiquitous computing environment of IoT connecting heterogeneous devices, technologies, and applications, and generating a

large number of events continuously brings in important and new challenges, such as interoperability, security, confidentiality, privacy, and energy-efficient operations [4]. For example, location tracking by the IoT devices may be allowed by some people to get personalized services; however, it may violate their privacy. The middleware, which is a software application, can hide the things details from the applications by communicating with the heterogeneous connected devices/things, filtering the raw captured data, and processing them before dissemination to the connected applications, and therefore easing the backend applications' development and offering multiple common services [5]. The middleware can also deal with the interoperability, security, and privacy issues facing the IoT. The IoT middleware development is an active research area; there exist many middleware solutions addressing the IoT environment requirements in terms of context awareness, scalability, interoperability across heterogeneous things, device management, data storage and management, security, privacy, and service deployment. A major challenge faced by application developers today is finding the most appropriate IoT middleware solution in terms of the provided functionalities that should meet the application requirements and the underlying used technologies. Therefore, the existing works on IoT middleware architecture need to be analyzed to address their existing technical challenges, issues, and gaps in this domain and suggest further improvements. This chapter provides a detailed overview of existing middleware solutions for IoT and is organized as follows: Section 2 provides background about IoT characteristics, architecture, and applications, and gives an overview of the IoT middleware general architecture. Section 3 presents the IoT middleware design considerations and requirements. Section 4 provides a comprehensive review of currently existing research work in designing IoT middleware platforms. Section 5 discusses criteria for choosing the right platform according to the application requirements, along with some open issues and challenges, and the last Section 6 provides some concluding comments recommending future research directions in this area.

2. Background

2.1 IoT architecture and applications

The Internet of Things (IoT) consists of two words: The "Internet" is defined "a network of networks and a global system of interconnected computer networks that use TCP/IP as a standard Internet Protocol (IP) to connect millions of users and multiple private, public, academic, business, and government networks," and "Things" include "any real-world object/physical element such as home appliances, clothes, smartphones, etc. or living things like people, animals, or plants" [6]. The International Telecommunication Union (ITU) considers IoT as "a worldwide network of interconnected objects, allowing anything and anyone to be connected, anytime and anyplace using any network and any service" [7]. Therefore, in IoT, many heterogeneous devices will be connected to the Internet and will provide a large volume of data and even services. The major components of IoT include wireless sensors and actuators networks, machine-to-machine (M2M) communications, and RFID/near-field communication (NFC) as shown in **Figure 1**.

2.1.1 IoT infrastructure characteristics

2.1.1.1 Heterogeneous intelligent devices

In IoT, heterogeneous devices in terms of features, capacities, sensor computing natures (high end, middle end, and low end), costs, embedded intelligence

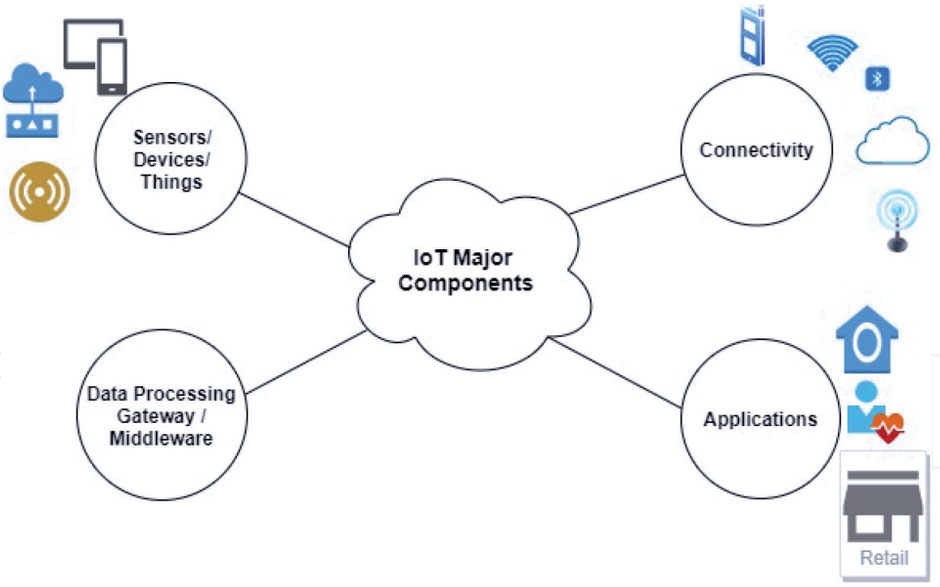

Figure 1.
IoT major components.

(adapting to the context, environment, and circumstances), and from different vendors are expected to communicate and exchange information [8]. Also, new types of devices are emerging continuously in the future as new technologies are developed [8]. **Figure 2** shows the main technologies used in IoT.

2.1.1.2 Context and location awareness

The different connected devices/things capture large volumes of data that need further processing; it should be filtered, interpreted, and put in a context to have a meaning. Context awareness helps to make the interpretation of data easier by adding context information to the raw data captured by the IoT things, which allows performing M2M communication that is considered a core element in an IoT environment [9].

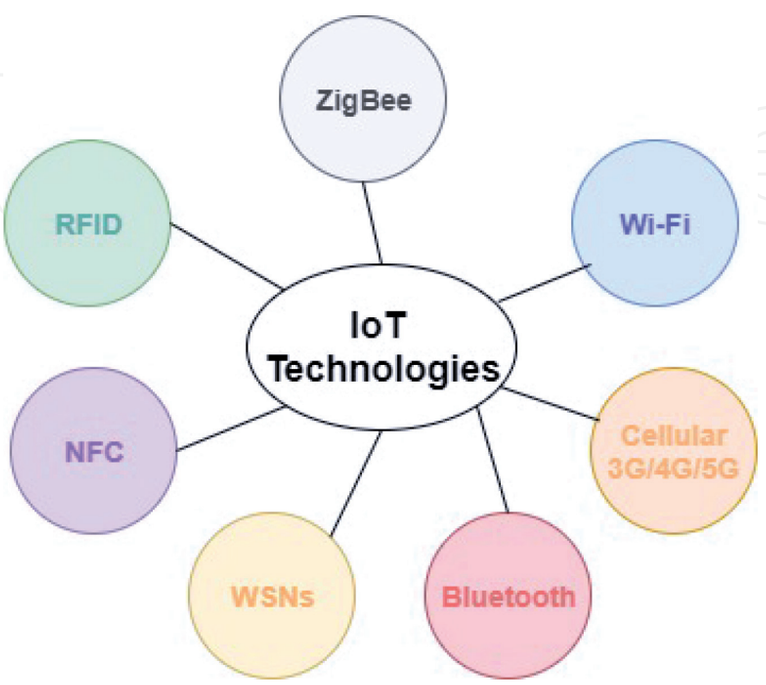

Figure 2.
IoT technologies.

Also, the spatial/location information about things is important to understand their interactions with other surrounding things (e.g., objects and people) [10].

2.1.1.3 Limited resources

IoT devices including small embedded sensors, RFID tags and readers, actuators, etc., are constrained in terms of processing, communication capacity, battery, and memory [8]. Also, the cost of these devices may increase when their performance increases in terms of processing, communication capacity, or the use of the battery to power them (e.g., active RFID tags are more expensive than the passive ones [5]).

2.1.1.4 Voluminous data and a continuous generation of spontaneous events

There are trillions of connected objects that are exchanging and storing hundreds of Exabytes of noisy data in IoT, and therefore forming an ultra-large-scale network [11]. These sudden interactions among things will also continuously generate events causing network congestion [11].

2.1.1.5 Dynamic distributed infrastructure

The IoT network is considered as an *ad hoc* network; there is no dedicated server for managing the resources of devices/things, and devices can join or leave the network anytime they want, or they can disconnect due to battery power shortage or connectivity problems. Cooperation between nodes will be needed to keep an active and stable network, and support multiple applications' development [11]. Therefore, the IoT network is considered a globally distributed network like the Internet and a local one within an application domain/context.

2.1.2 IoT applications characteristics

2.1.2.1 Diverse application domains

The IoT applications can be developed to cater to the needs of different domains and environments, having different requirements and deployment architectures, such as logistics and supply chain management, healthcare, environmental monitoring, smart home/buildings, smart agriculture [6]. **Figure 3** gives an overview of the potential IoT applications.

2.1.2.2 Real-time delivery of data and services

IoT applications in some specific domains such as transportations and healthcare need to communicate real-time data and deliver on-time services to avoid critical situations [6].

2.1.2.3 Security and privacy concerns

In the IoT network, the security of applications and communications among the different nodes should be considered, along with the privacy of people's captured data such as location, daily activities, buying habits [12]. An efficient and scalable security mechanism should be implemented considering the *ad hoc* nature of the IoT network, and also, the privacy issues should be considered not to prohibit the deployment of applications that violate citizen's privacy by the law [12].

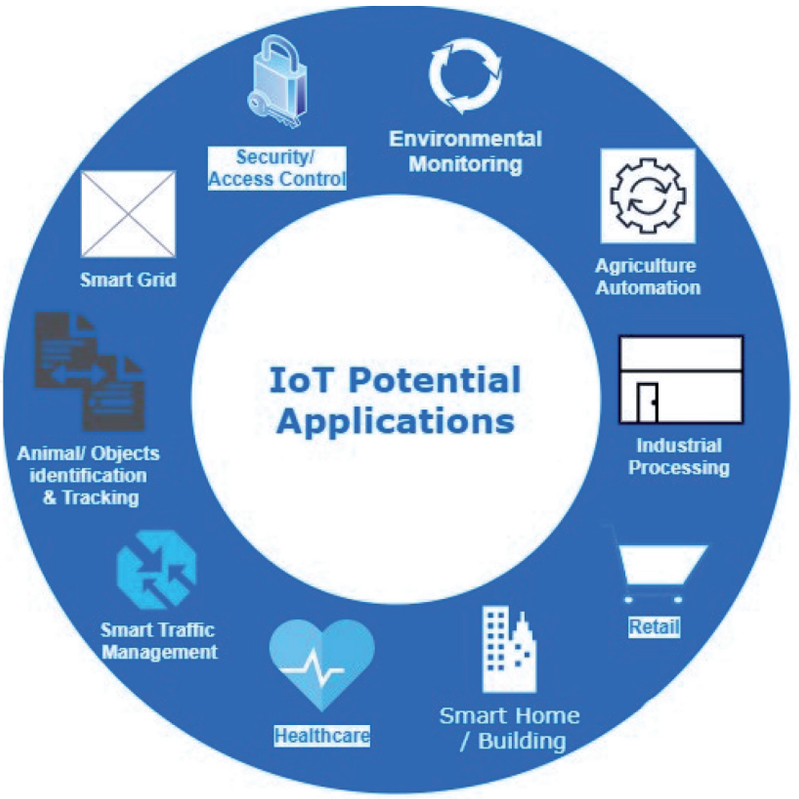

Figure 3.
IoT potential applications.

2.2 IoT middleware platform general architecture

Given the IoT infrastructure and applications' characteristics stated above, and based on my previous research work done on middleware architecture for RFID [5], context aware, and ubiquitous computing [13], an IoT middleware solution can generally provide the following functionalities:

- Device abstraction, discovery, management, and control: It includes interoperation among the heterogeneous connected devices/things using different standards. Application programming interfaces (APIs) are used for abstracting the communication with the physical layer and also for disseminating data and services to the different connected backend applications, hiding all details and complexities.

- Data management and dissemination: It provides the different data preprocessing functionalities, such as filtering, duplicate removal, aggregation.

- Context detection and processing

- Security, privacy, and business rules processing

- Application abstraction

The IoT middleware architecture is depicted in **Figure 4**. The main layers include device abstraction and resource management layer, which handle the interoperability and interaction with the heterogeneous devices, and manage the low-level hardware parameters such as the used protocols, communication technologies, standards, and air interface; data management layer is responsible for storing

Figure 4.
IoT middleware architecture.

and processing (filtering, aggregation, inference, etc.) the raw data captured by the different devices/things; event management and context detection layer include the application of policies and business rules requested by the applications (e.g., security and privacy rules); and application abstraction layer allows the communication of applications with the different devices and helps them to get the desired processed data and generated events from the middleware.

3. IoT middleware design considerations and requirements

The role of a middleware platform is to provide a software layer shielding the complexities of the hardware layer including the operating systems from the applications and allowing the applications' developers to be concentrated mainly on the requirements/problem to be solved. As described in Section 2, in the context of IoT, there is a considerable variation in the used technologies, standards, and network communications. We describe herein, a set of design considerations and requirements for a middleware to suit the IoT infrastructure and application characteristics.

3.1 Resource discovery and management

Since the IoT infrastructure is dynamic by its nature, the IoT middleware should provide an automatic device discovery and enable the IoT heterogeneous hardware devices (e.g., RFIDs, sensors, smartphones) to detect their neighbors in the network and show their presence and available resources to them. In this case, the middleware should consider the characteristics of the resource-constrained IoT devices and be scalable in terms of the number of connected devices in the network. The middleware should also manage the devices, monitor their resource usage, and resolve any resource conflicts when potential and spontaneous new devices are connected to satisfy the application requirements.

3.2 Data management, context awareness, and event management

The IoT middleware should provide data management and processing functionalities to the backend applications; these include but are not limited to data detection and acquisition from the different connected devices/things, data preliminary

processing, such as filtering, duplicate removal, compression, aggregation, and data storage. The IoT middleware should also manage the high number of generated events in an IoT environment, such as real-time dissemination of events to the applications, event transformation based on contextual/location data, and inferences.

The IoT-middleware should provide context detection and processing for it to adapt to smart applications requirements; it should collect context data and then process them to generate inferences and decisions. This could be achieved by using different techniques such as knowledge database, data mining algorithms, semantic context aware multimodal visualization approach, and the use of optimized message communication between the middleware users.

3.3 Scalability and adaptability

The IoT network can include a large number of connected things/devices and provide multiple services; therefore, the IoT middleware should be scalable allowing the growth of the IoT network, including the emergence of new heterogeneous devices that could be monitored, added, or removed without any impact on existing middleware functionalities, the provision of new services/functionalities, the addition/removal of network nodes, and the connection of multiple interesting applications in the middleware services without complexity. The use of IPv6, loose coupling, and virtualization are considered as useful ways for improving scalability in middleware solutions. Also, the use of a service-oriented architecture (SOA) makes the middleware flexible to the applications' requirements in terms of new services. The IoT middleware should also be dynamically adaptive to the different circumstances and changes in the IoT environment.

3.4 Real-time data capture and services

The IoT network deals with multiple real-time/time-critical applications requiring a timeliness delivery of processed data and services without any delay, for example, healthcare applications; therefore, the middleware should provide real-time services and information to these applications. In this case, the middleware should manage the large data volumes detected from the multiple connected devices and therefore use novel methods to detect, process, and disseminate these data to the interested applications. The challenge of transaction handling, indexing, and querying these data should also be handled. This could be ensured through the use of agents, query processors, notification managers, etc.

3.5 Reliability and availability

Every component or layer in the IoT middleware should be operational including communication, data processing, events management, technologies, devices connectivity, and application management, even when failures occur. It should provide a stable service for applications/users even at times of failure. The middleware must also be available at all times for mission-critical applications that require a high fault tolerance, for example, medical applications; in the case of failure, the recovery time should be reduced to cater to the applications' availability requirements.

3.6 Security and privacy

The IoT middleware should consider the security and privacy rules and policies required by the connected applications. The use of context awareness in the middleware can disclose some personal information about individuals such as location;

therefore, it needs to protect people's privacy using policies/rules/ontologies depending on the applications' specific needs [12]. Also, most of IoT middleware solutions are evolving into the cloud, which requires more mechanisms to be put in place to deal with the security and privacy issues, making users safe and protecting their personal data.

3.7 Ease of use and deployment

The IoT middleware should be lightweight, and easily used and deployed by the end-users of devices or applications without any complicated setup procedures.

3.8 Distributed implementation

If the IoT infrastructure is distributed, the middleware implementation should also be distributed, for example, when the devices, applications, and users are located in different geographical areas.

Some of the requirements stated above are considered to be mandatory for some applications while optional for others; for example, the real-time data capture and services are highly required in the case of medical applications, but it is optional for other applications that do not use timeliness information. However, the security, privacy, and interoperability functionalities are strictly required by all types of applications.

4. Overview of existing IoT middleware solutions

Many middleware solutions, using a single design approach (e.g., service-based, agent-based, database-based) or a hybrid one (combining different design approaches), and providing different functionalities in many application domains have been proposed and implemented in the IoT. These initiatives aim to offer a standard platform used to abstract the lower-level details of the connected physical devices and offer multiple services to the users and/or applications. In this chapter, the existing IoT middlewares are surveyed based on their used design approach and are grouped into six categories: service-oriented middleware, agent-based middleware, event-based middleware, virtual machine-based middleware, database-oriented middleware, and application-oriented middleware. A comparison of these design approaches is given in **Table 1**.

4.1 Service-oriented middleware solutions

The service-oriented middleware (SOM), based on the service-oriented design pattern, provides services to the applications, such as service discovery and management, data management, and quality of service (QoS) management. There exist many service-oriented IoT middleware solutions. Some of the commonly used service-oriented IoT middleware solutions are described as follows:

Hydra is a SOM for ubiquitous computing providing many management components for resources, security, and services [19]. Hydra is a lightweight middleware supporting dynamic self-reconfiguration and optimizing energy consumption in battery-constrained devices. The security and privacy requirements are ensured by Hydra through the use of Web Services enriched by semantic resolution [20].

The *SenseWrap* [21] middleware solution uses virtual sensors with the Zeroconf protocols to abstract the sensors' low-level details from the applications, and allow them to discover sensor-hosted services. This middleware solution applies virtualization only to sensors, which makes it unsuitable for IoT environments including heterogeneous devices and application domains.

			IoT middleware requirements/features									
Middleware approach	**Middleware solutions**	**Target environment**	**Interoperability**	**Scalability**	**Adaptability**	**Real timeliness**	**Security and privacy**	**Reliability**	**Context awareness**	**Ease of use and deployment**	**Data management**	**Event management**
Service-oriented	**Hydra**	WSNs	Yes	Yes	No	Yes	Yes	No	Yes	Yes	Yes	Yes
	SenseWrap	Virtual sensors	Yes	Yes	No	No	No	No	No	Yes	No	Yes
	MUSIC	Ubiquitous	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No	Yes
	SENSEI	Sensors/ actuators	No	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
	TinySOA	WSNs	No	Yes	Yes	Yes	No	No	No	Yes	No	No
	SensorsMW	WSNs	No	Yes	Yes	Yes	No	No	No	Yes	Yes	No
	Servilla	WSNs	No	Yes	Yes	Yes	Yes	Yes	No	Yes	No	No
	SOCRADES	Heterogeneous devices	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	Middleware based on REST API	Heterogeneous devices	Yes	Yes	Yes	Yes	No	No	No	Yes	No	Yes
	3SOA	IoT	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
Cloud-based Serivce-oriented	**Google** Fit	IoT, cloud	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No
	Xively	IoT, cloud	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes
	CarrIoTs	IoT, cloud	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	Echelon	IoT, cloud	Yes	Yes	Yes	Yes	Yes	No	No	Yes	Yes	No

			IoT middleware requirements/features									
Middleware approach	**Middleware solutions**	**Target environment**	**Interoperability**	**Scalability**	**Adaptability**	**Real timeliness**	**Security and privacy**	**Reliability**	**Context awareness**	**Ease of use and deployment**	**Data management**	**Event management**
Microservices-based	**Arrowhead Framework** [14]	IoT	Yes	Yes	Yes	Yes	Yes	No	No	Yes	Data exchange	Yes
	General microservice architecture [15]	IoT	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
	Smart City [16]	IoT	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
	Ocean [17]	IoT	Yes	Yes	Yes	yes	No	No	Yes	Yes	Yes	No
	Web of Objects Architecture [18]	IoT	Yes	Yes	Yes	Yes	No	No	Yes	yes	Yes	No
Agent-based	**Impala**	WSNs	No	No	No	Yes	No	No	No	Yes	No	Yes
	ActorNet	WSNs	No	No	No	Yes	No	Yes	Yes	Yes	Yes	Yes
	Agilla	WSNs	Yes	No	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes
	Ubiware	IoT, ambient	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes
	Smart messages	WSNs, Embedded Systems	No	No	No	Yes	Yes	No	Yes	Yes	No	No
	ACOSO-based middleware	IoT	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes

			IoT middleware requirements/features									
Middleware approach	**Middleware solutions**	**Target environment**	**Interoperability**	**Scalability**	**Adaptability**	**Real timeliness**	**Security and privacy**	**Reliability**	**Context awareness**	**Ease of use and deployment**	**Data management**	**Event management**
Event-based	**EMMA**	IoT, Cloud	Yes	Yes	Yes	Yes	No	Limited	No	Yes	No	Yes
	Hermes	Large-scale distributed and ubiquitous systems	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No	Yes
	Event-based Middleware for Syntactical Interoperability in IoT	IoT	Yes	Yes	Yes	Yes	No	Yes	No	Yes	No	Yes
Virtual-machine-Based	Maté	WSNs	No	Yes	Yes	Yes	No	No	Yes	Yes	No	No
	VM*	WSns	No	Yes	No	Yes	No	No	No	Yes	No	Yes
	Melete	WSNs	No	Yes	Yes	Yes	No	Yes	No	Yes	No	Yes
Database-oriented	**SINA**	WSNs	No	Yes	No	Yes	No	No	No	Yes	Limited	Yes
	IrisNet	WSNs	No	Yes	No	Yes	No	No	No	Yes	Yes	No
	Sensation	WSNs	No	Yes	No	Limited	No	No	No	No	Yes	No
	TinyDB	WSNs	No	No	No	Limited	No	No	No	No	Yes	No
	HyCache	WSNs	No	No	No	Limited	No	No	No	No	Yes	No

			IoT middleware requirements/features									
Middleware approach	**Middleware solutions**	**Target environment**	**Interoperability**	**Scalability**	**Adaptability**	**Real timeliness**	**Security and privacy**	**Reliability**	**Context awareness**	**Ease of use and deployment**	**Data management**	**Event management**
Application-oriented	AutoSec	Distributed	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes
	Adaptive middleware	WSNs/ Healthcare	No	Yes	Yes	Yes	No	Yes	Yes	Yes	No	Yes
	MlLAN	WSNs/ Healthcare	No	Yes	Yes	Yes	No	No	Yes	Yes	No	Yes
	MidFusion	WSNs/ Information Fusion	No	Yes	Yes	Yes	No	No	No	Yes	Yes	Yes
	TinyCubus	Driver Assistance Systems	Yes	Yes	Yes	Yes	No	No	No	Yes	Yes	No

Table 1.
Comparison of existing IoT middleware solutions.

The *MUSIC* middleware [22] supports building systems in ubiquitous environments where service providers and consumers may change dynamically based on context. Its architecture is composed of different managers providing different functionalities, including the context manager, service discovery manager, QoS manager, SLA monitoring, and adaptation manager. The use of context data by the MUSIC middleware may increase the risk of privacy leakage in an IoT environment. *SENSEI* [23] is another middleware solution including context services and a context model for the real world Internet including IoT. Its resources use ontologies for their semantic modeling, which makes it unsuitable for large-scale IoT networks since there are no standard established ontologies yet.

TinySOA [24] is a service-oriented middleware used for WSN applications development. It provides a management of WSN devices and communications, and allows applications to get processed data from the connected sensors. TinySOA allows only a few functionalities, such as abstraction and resource discovery related to WSNs, and does not support other devices; therefore, it could not be used fully within an IoT network [24]. Another SOM providing the management of quality of service in WSNs is called *SensorsMW* [25]. *Servilla* middleware also facilitates application development using heterogeneous WSNs; however, it is not widely used due to the privacy and security threats caused by the individual sensor-level access [26].

SOCRADES middleware [27] contains a layer for devices and services monitoring responsible for devices/things management and service discovery, and another one for application services such as event storage. The middleware provides a security solution by using authentication to control access to the different devices. However, the privacy of sensitive information is not ensured, since a direct access to the connected devices and their offered services is allowed by the middleware.

There exist many other cloud-based service-oriented IoT middleware solutions, such as *Google Fit*, *Xively*, *CarrIoTs*, *Echelon*; however, there are still many concerns about the cloud platform security and privacy, especially for mission-critical IoT applications [28].

Recent studies have been conducted concerning the design and implementation of service-oriented IoT middleware solutions including the one in [29] that suggests a middleware based on REST API to collect data from different devices, intending to deal with the heterogeneity issues. The authors in [30] presented a 3SOA (Sensing-as-a-Service run-time Service-Oriented Architecture) middleware solution that allows interoperability among IoT platforms, and highly abstracts the applications from the low-level details of IoT hardware platforms and communication languages.

In conclusion, most old SOMs manage only WSNs and do not scale to the use of multiple heterogeneous devices as in the context of IoT. Recent suggested service-based middleware platforms provide solutions for the interoperability and heterogeneity problems; however, they still offer a limited security through the use of authentication, do not use unified service standards, and require automation for service configuration and optimization due to the recurring demands of new services by the interesting applications.

Another type of microservices-based architecture has been recently proposed to develop IoT platforms that meet the heterogeneous and distributed nature of IoT devices, and provide dynamic, scalable, maintainable, and interoperable IoT environments. Delsing et al. [14] propose an Arrowhead Framework architecture enabling scalability, security, and real-time performance in a multi-cloud setting. This architecture supports multiple IoT devices based on SOA architecture in local clouds to exchange inter- and intra-cloud information, and allows organizations to move toward a multi-stakeholder cooperation catering to market requirements and supporting efficiency, flexibility, and sustainability [14].

A general microservice architecture for IoT applications development is proposed by Sun et al. [15], providing flexibility, scalability, maintainability, light-weightness, and loose coupling to deal with the different challenges of the continuous IoT development. The authors focus on the system design based on microservices and device communication protocols used between the service layer and physical device layer. This framework allows, therefore, more interoperability, automation, and intelligence and provides big data and geo-localization services [15].

Another recent architecture based on microservices is proposed by Lai et al. [16] to provide IoT services for multi-mobility in a smart city. The architecture provides flexibility and scalability to efficiently manage the different heterogeneous IoT devices using independent microservices, which could be separately deployed in a distributed system [16]. The authors used real-case scenarios to test the architecture using multi-mobility services for citizens in a smart city.

A recent study [17] also shows how the use of a framework based on microservices allows to mitigate the critical challenges of IoT devices and applications, and increases their scalability when deployed in the ocean where there is a continuous increasing growth of big data.

Many other microservices-based IoT platforms have been proposed in various application domains such as smart farms [31], smart logistics/factories [32], smart cars [33], and smart commerce [34]. Jarwar et al. [18] also proposed a cross-domain/general-purpose Web of Objects Architecture for IoT service provisioning in which a virtual object is used as an abstraction of a physical object.

4.2 Agent-based middleware solutions

Agent-based middleware solutions use mobile agents to facilitate distribution throughout the network and allow a partial failure tolerance. The use of mobile agents in the IoT network provides many advantages including interoperability with the heterogeneous devices, reliability and availability, resource and code management taking into consideration the resource-constrained devices, and application management. Some of the most commonly used agent-based middleware solutions are highlighted below.

Impala [35] is an agent-based middleware solution enabling code management, application modularity, resource management, mobility, and openness in WSNs. Its architecture also allows an improvement of the efficiency of resource-constrained nodes. However, Impala middleware does not provide the raw data cleaning functionality, which is necessary for an IoT setting.

Other examples of agent-based WSN middleware solutions include *ActorNet* [36] that provides context management and allows application development taking into consideration the limited resources in a WSN environment. However, ActorNet uses a service discovery mechanism leading to a slow network. *Agilla* [37] is another example of agent-based platforms, which deploys independent event-related mobile agents in every sensor node; however, this is limited due to the constrained resources of nodes, which may cause message loss and interference with programmability and code management tasks.

Ubiware [38] is considered a dedicated agent-based middleware solution for IoT, which supports resource discovery, invocation, monitoring, and the development of multiple extensible applications. Ubiware is a Java-based solution with a three-layer architecture where resources are interpreted as Java components; it uses ontologies and policies to satisfy the security and interoperability requirements; however, these policies do not include all the available WSN standards. There exist many other Java-based middleware solutions dedicated to WSN applications, such as *AFME*, *MAPS*, *MASPOT*, and *TinyMAPS* to name a few [39].

Smart messages [40] middleware is a highly flexible solution for dynamic network configurations; it overcomes the limitations of volatile, heterogeneous, and resource-constrained embedded systems using agent migration. However, it is limited in terms of the number of connected applications and its support to multiple devices in the case of an IoT context.

The authors in [41] present a new approach for increasing the smart objects' self-adaptation and allowing them to make autonomous decisions and be smarter based on a multi-agent system (MAS). The authors in [42] also presented a new multi-agent-based approach called ACOSO (Agent-based Cooperating Smart Objects) and its related middleware catering for the heterogeneous IoT platforms. The flexibility and effectiveness of this middleware were proved through the implementation of a "Smart University system."

The autonomous behavior of agents used in middleware solutions may lead to the IoT network's self-organization and fault tolerance. However, the dynamic behavior of agents may lead to message loss; therefore, most of the above-discussed middleware solutions could not be used within the large-scale IoT networks requiring a heterogeneous infrastructure, including resource-constrained devices.

4.3 Event-based middleware solutions

All the components of an event-based middleware solution use a publish/subscribe model; the event sending component is called the producer or publisher, and the receiving component is called the consumer or subscriber. The consumers are registered for a particular event published by the producers for which they are frequently receiving notifications. The event-based approach provides timeliness, security, scalability, availability, reliability, and fault tolerance.

EMMA [43] is an available Java Message Service middleware, which is a type of event-based approach designed for video communication systems to provide many types of messaging. However, it is not energy efficient and provides only a limited reliability.

Hermes middleware [44] also provides scalability, interoperability, and reliability, and it is also fault tolerant. However, it provides only a limited adaptation and does not allow a composite and persistent storage of events.

The authors in [45] proposed an event-based middleware solution implemented using the publish-subscribe pattern to solve the problem of interoperability in IoT. The interoperability assessment methodology was used to test the middleware performance, and it was shown that it is qualified compared to previous systems.

There exist many other event-based middleware solutions including *GREEN* [46], *RUNES* [47], *Steam* [48], *PSWare* [49], *PRISMA* [50], and *TinyDDS* [51], which are appropriate for systems involving a high mobility and failure occurrence. However, they do not adequately address the context awareness, adaptability, interoperability, security, privacy, and timeliness requirements of the IoT. Also, the concurrency of the event in this type of middleware solutions may lead to reduced system reliability.

4.4 Virtual machine-based middleware solutions

The virtual machine (VM) middleware approach considers virtualizing the network infrastructure, where the different network nodes are holding a VM and applications are designed as separate modules distributed throughout the network. This ensures self-management, and a high level of abstraction and adaptability. *Maté* [52] is a middleware solution based on VM, which addresses the different challenges in WSNs and is designed for nodes with limited energy and bandwidth

resources. Mate is based on a VM approach and provides byte code interpretation and tackles the different challenges in WSNs; however, it does not provide event management and does not allow a single sensor node to support multiple applications. Some other middleware solutions based on the VM approach were built on top of Mate to extend its capabilities, including *VM** [53] and *Melete* [54]. These provide resource management, code dissemination, and an easy concurrent application deployment; however, they do not handle a dynamic network topology.

There exist some middleware solutions based on Java virtual machine (JVM), such as *MagnetOS*, *Squawk*, and *Sensorware* which allows them to support multiple portable applications; however, they are unsuitable for the IoT resource-constrained devices since they use heavy mechanisms for interlayer communication and computation consuming memory and processing power [55]. These constraints make the VM-based approach suitable only for resource-rich devices.

The application-specific virtual machine (ASVM) approach has been developed to target specific application domains. Middleware solutions based on this approach include but are not limited to *TinyVM* [56], *SwissQM* [57], and *TinyReef* [58]. However, the ASVM approach is still heavyweight, which makes it unsuitable for the limited-resource devices in an IoT network deployment.

4.5 Database-oriented middleware solutions

The whole network in this type of middleware solution is viewed as a relational database, managed using a query language like SQL. For example, the *Sensor Information Networking Architecture (SINA)* middleware [59] enables applications to send queries, collect results, and monitor network changes in a WSN setting. It also supports resource management and monitoring, event monitoring, data preprocessing, while clustering sensor nodes to ensure scalability and energy-efficient operations. However, SINA is not context aware, and it does not support security, privacy, and interoperability. *IrisNet* [60] is another distributed and lightweight database-oriented middleware solution providing simultaneous heterogeneous WSN services using queries over the collected data from the sensor nodes. However, it does not resolve the issues related to energy efficiency, interoperability, adaptiveness, and context awareness. Other examples of database-oriented middleware solutions include *Sensation* [61], *TinyDB* [62], and *HyCache* [63]. In these solutions, database queries are used to get approximate data of interest from the sensor nodes; they do not support the real-time requirement of the IoT infrastructure. They are also energy inefficient and use a centralized model, which does not scale to the ultra-large dynamic IoT networks [59]. Also, they do not provide the data aggregation and knowledge discovery functionalities.

4.6 Application-oriented middleware solutions

Application-oriented middleware solutions are dedicated to specific domain requirements and infrastructure. For example, the *Automatic Service Composition (AutoSec)* middleware supports one application at a time using resource provisioning and information collection policies set by the different applications [64]. *Adaptive middleware* is designed for smart home applications providing context awareness, and it also supports adaptation for other applications and ensures the quality of information collection and transmission between the network nodes [65]. Other examples include *MlLAN* middleware [59] that targets the healthcare applications and adapts to their QoS requirements at runtime, *MidFusion* [66] designed for information fusion applications such as intrusion detection systems,

and *TinyCubus* [59] designed for driver assistance systems that satisfies the application requirements by customizing its generic components.

The application-specific approach leads to the design of special-purpose middleware systems dedicated to a specific application domain, using a centralized mechanism for resource discovery. This makes them unsuitable for the distributed and fault-tolerant nature of IoT environments.

4.7 Hybrid approach middleware solutions

There exist some middleware platforms using a hybrid approach, combining two or more design approaches stated above. For example, both *SOCRADES* [27] and *Servilla* [26] service-oriented middleware solutions use also the virtual machine (VM)-based approach. The VM in Servilla, for example, serves to execute application tasks, while the service provisioning framework (SPF) (the service-oriented part) is used to discover and execute services on individual sensor nodes in a WSN. A middleware solution designed for the manufacturing domain using the hybrid approach is also proposed in [67], taking the advantages of both the database-oriented and semantic modeling approaches for ensuring an accurate and efficient data management and communication among the different devices and applications.

Table 1 shows the IoT requirements/features available in each middleware design approach and provides a comparison of the different IoT middleware solutions described in Section 4. The choice of the comparison criteria is based on the works cited above, from which the most common, essential, critical, and important characteristics that are shared between the different IoT platforms have been extracted. The description of each criterion is given above in Section 3 (IoT Middleware Design Considerations and Requirements). There exist many additional/non-functional criteria and features, which could be available in some IoT platforms such as recoverability, fault-tolerance, maintainability, configurability, mobility, reusability. But these are not subject of this review since it targets only the most essential design features/functionalities of IoT middleware solutions.

5. Open issues in IoT middleware design

According to the previous comparison, most of the works concentrate their efforts on providing basic functionalities such as ease of deployment, data management, event management, and real-timeliness. A considerable effort must be made in interoperability and adaptability, which allows devices/things using heterogeneous protocols to connect. Context awareness is also a feature that is not considered by most of the described middleware solutions and still encounters many shortcomings. In addition, security and privacy features need particular attention from researchers, because they are missed in almost all the reviewed middleware solutions above.

In summary, the most challenging issues that still persist in IoT-middleware design, implementation, and deployment are listed below:

- Standardization: The use of heterogeneous devices within a variety of application domains in the IoT makes the use of a single standard for a middleware solution impossible. However, many research works tend to implement a standardized middleware solution for a specific domain, such as semantic web applications domain, sensor networking environments, and smart offices [59, 65]. This will allow application developers to select a middleware solution following the desired standard within a certain domain.

- Storage capacity: The storage capacity of the heterogeneous connected things within the IoT should be considered when implementing a middleware solution. For example, if the middleware solution offers many services and data management functions, it will be difficult to use it with low-level storage devices. This issue could be addressed by defining storage requirements by the different types of backend applications, taking into consideration the application domain, before choosing an adequate middleware solution.

- Security and privacy: IoT middleware solutions can rely on a single layer for providing security and privacy to the backend applications, or distribute the security and privacy support among all the middleware layers. Either way, security and privacy support will add more processing overhead to the middleware platform, and it should also take into consideration the security and privacy requirements and rules for each specific application with minimum overhead.

- Applications abstraction: The IoT middleware should include an application abstraction layer to allow multiple backend applications to be registered with the middleware, and to specify the set of services and data processing functions needed. The applications can also specify policies/rules concerning some functionalities, such as context awareness, security, privacy, data processing, and event processing and inferences.

6. Conclusion and future work

Middleware is becoming a necessity for managing heterogeneous devices in the IoT network and developing applications in different domains. There exist a variety of middleware platforms designed for IoT. This chapter provides a detailed overview of existing IoT middleware solutions, and discusses the technical challenges and open issues involved in designing these platforms including device and application abstraction, scalability, context awareness, event management, unfixed infrastructure, security, and privacy. In future work, the open issues in IoT could be further investigated to suggest possible new approaches to solve them. Also, a new middleware design approach may be proposed to include a new perspective for managing the IoT devices/things and applications, including a solution for the unexplored open issues in a specific application domain, such as security, privacy, and interoperability. A test of this new approach could be performed using my previous proposed middleware solution for RFID described in [5].

IntechOpen

Author details

Mehdia Ajana El Khaddar
Alakhawayn University, Ifrane, Morocco

*Address all correspondence to: mehdia.ajana@gmail.com

References

[1] Mahmoud Elkhodr M, Shahrestani S, Cheung HS. Internet of Things applications: Current and future development. In: Hassan QF, editor. Innovative Research and Applications in Next-Generation High Performance Computing. 1st ed. Hershey, Pennsylvania: IGI Global; 2016. pp. 397-427. DOI: 10.4018/978-1-5225-0287-6.ch016

[2] GLOBE NEWSWIRE. Internet of Things (IoT) Market—Growth, Trends, Forecasts (2020-2025) [Internet]. 2020. Available from: https://www.globenewswire.com/news-release/2020/05/13/2033070/0/en/The-global-IoT-market-is-expected-to-reach-a-value-of-USD-1256-1-billion-by-2025-from-USD-690-billion-in-2019-at-a-CAGR-of-10-53-during-the-period-2020-2025.html [Accessed: 01 March 2021]

[3] GSMA. IoT Connections Forecast: The Rise of Enterprise [Internet]. 2019. Available from: https://www.gsma.com/iot/resources/iot-connections-forecast-the-rise-of-enterprise/ [Accessed: 21 February 2021]

[4] Jason IH, James AL. Four Technological Challenges in Ubiquitous Computing and their Influence on Interaction Design [Internet]. Available from: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.419.5005&rep=rep1&type=pdf [Accessed: 21 February 2021]

[5] Ajana ME, Boulmalf M, Harroud H, Elkoutbi M. RFID middleware design and architecture. In: Turcu C, editor. Designing and Deploying RFID Applications. Rijeka: InTechOpen; 2011. DOI: 10.5772/16917. ISBN: 978-953-307-265-4. Available from: http://www.intechopen.com/books/designing-and-deploying-rfid-applications/rfid-middleware-design-and-architecture

[6] Ajana ME, Boulmalf M. Smartphone: The ultimate IoT and IoE device. In: Mohamudally N, editor. Smartphones from an Applied Research Perspective. Rijeka: IntechOpen; 2017. DOI: 10.5772/intechopen.69734. Available from: https://www.intechopen.com/books/smartphones-from-an-applied-research-perspective/smartphone-the-ultimate-iot-and-ioe-device

[7] Gopalsamy BN. Communication trends in Internet of Things. In: Sugumaran V, editor. Developments and Trends in Intelligent Technologies and Smart Systems. 1st ed. Hershey, Pennsylvania: IGI Global; 2018. pp. 248-305. DOI: 10.4018/978-1-5225-3686-4.ch014

[8] Yacchirema Vargas DC, Palau Salvador CE. Smart IoT gateway for heterogeneous devices interoperability. IEEE Latin America Transactions. 2016;**14**(8):3900-3906. DOI: 10.1109/TLA.2016.7786378

[9] Ntalasha D, Renfa L, Wang Y. Internet of thing context awareness model. EAI Endorsed Transactions on Context-aware Systems and Applications. 2016;**3**(7):151084. DOI: 10.4108/eai.12-2-2016.151084

[10] Cristea V, Dobre C, Pop F. Context-aware environments for the Internet of Things. In: Bessis N, Xhafa F, Varvarigou D, Hill R, Li M, editors. Internet of Things and Inter-cooperative Computational Technologies for Collective Intelligence. USA: Springer; 2013. pp. 25-49. DOI: 10.1007/978-3-642-34952-2_2

[11] Krishnamurthi R, Kumar A, Gopinathan D, Nayyar A, Qureshi B. An overview of IoT sensor data processing, fusion, and analysis techniques. Sensors. 2020;**20**(21):6076. DOI: 10.3390/s20216076

[12] Oorschot P C Van, Smith S W. The Internet of Things: Security challenges.

IEEE Security & Privacy. 2019;**17**(5):7-9. DOI: 10.1109/MSEC.2019.2925918

[13] Ajana ME, Chraibi M, Harroud H, Boulmalf M, Elkoutbi M, Maach A. FlexRFID: A security and service control policy-based middleware for context-aware pervasive computing. International Journal of Advanced Research in Artificial Intelligence (IJARAI). 2014;**3**(10):26-34. DOI: 10.14569/IJARAI.2014.031004

[14] Delsing J et al. The arrowhead framework architecture: Arrowhead framework. In: Delsing J, editor. IoT Automation. United States: CRC Press Publisher; 2017. DOI: 10.1201/97813 15367897-4. ISBN: 9781498756754

[15] Sun L, Li Y, Memon RA. An open IoT framework based on microservices architecture. China Communications. 2017;**14**(2):154-162. DOI: 10.1109/ CC.2017.7868163

[16] Lai C, Boi F, Buschettu A, Caboni R. IoT and microservice architecture for multimobility in a smart city. In: Proceedings of the IEEE 7th International Conference on Future Internet of Things and Cloud (FiCloud); 26-28 August 2019; Istanbul, Turkey. New York: IEEE; 2019. pp. 238-242. DOI: 10.1109/FiCloud.2019.00040

[17] Razzaq A. Microservices architecture for IoT applications in the Ocean: Microservices architecture based framework for reducing the complexity and increasing the scalability of IoT applications in the Ocean. In: Proceedings of the 20th International Conference on Computational Science and Its Applications (ICCSA); 1-4 July 2020; Cagliari, Italy. New York: IEEE; 2020. pp. 87-90. DOI: 10.1109/ICCSA50381. 2020.00025

[18] Jarwar MA, Kibria MG, Ali S, Chong I. Microservices in web objects enabled IoT environment for enhancing reusability. Sensors. 2018;**18**(2):352. DOI: 10.3390/s18020352

[19] Eisenhauer M, Rosengren P, Antolin P. HYDRA: A development platform for integrating wireless devices and sensors into ambient intelligence systems. In: Giusto D, Iera A, Morabito G, Atzori L, editors. The Internet of Things. New York: Springer; 2010. pp. 367-373. DOI: 10.1007/978-1-4419-1674-7_36

[20] Reiners R, Zimmermann A, Jentsch M, Zhang Y. Automizing home environments and supervising patients at home with the hydra middleware: Application scenarios using the hydra middleware for embedded systems. In: Proceedings of the First International Workshop on Context-aware Software Technology and Applications; 24 August 2009; Amsterdam, The Netherlands. New York: ACM; 2009. pp. 9-12. DOI: 10.1145/1595768.1595772

[21] Zgheib R, Conchon E, Bastide R. Semantic middleware architectures for IoT healthcare applications. In: Ganchev I, Garcia N, Dobre C, Mavromoustakis C, Goleva R, editors. Enhanced Living Environments. Cham: Springer; 2019. pp. 263-294. DOI: 10.1007/978-3-030-10752-9_11

[22] Rouvoy R, et al. MUSIC: Middleware support for self-adaptation in ubiquitous and service-oriented environments. In: Cheng BHC, de Lemos R, Giese H, Inverardi P, Magee J, editors. Software Engineering for Self-Adaptive Systems. Berlin: Springer; 2009. pp. 164-182. DOI: 10.1007/978-3-642-02161-9_9

[23] Tsiatsis V et al. The SENSEI real world internet architecture. In: Georgios T, et al., editors. Towards the Future Internet—Emerging Trends from European Research. Amsterdam, The Netherlands: IOS Press; 2010. pp. 247-256. DOI: 10.3233/978-1-60750-539-6-247

[24] Avilés-López E, García-Macías JA. TinySOA: A service-oriented

architecture for wireless sensor networks. Service Oriented Computing and Applications. 2009;**3**:99-108. DOI: 10.1007/s11761-009-0043-x

[25] Anastasi G F, Bini E, Lipari G. Extracting data from WSNs: A service-oriented approach. In: Anastasi G, Bellini E, Di Nitto E, Ghezzi C, Tanca L, Zimeo E, editors. Methodologies and Technologies for Networked Enterprises. Berlin: Springer; 2012. p. 329-356. DOI: 10.1007/978-3-642-31739-2_17

[26] Chien-Liang F, Gruia-Catalin R, Chenyang L. Servilla: A flexible service provisioning middleware for heterogeneous sensor networks. Science of Computer Programming. 2012;**77**(6):663-684. DOI: 10.1016/j.scico.2010.11.006

[27] de Souza LMS, Spiess P, Guinard D, Köhler M, Karnouskos S, Savio D. SOCRADES: A web service based shop floor integration infrastructure. In: Floerkemeier C, Langheinrich M, Fleisch E, Mattern F, Sarma SE, editors. The Internet of Things. Berlin: Springer; 2008. pp. 50-67. DOI: 10.1007/978-3-540-78731-0_4

[28] Ngu AH, Gutierrez M, Metsis V, Nepal S, Sheng QZ. IoT middleware: A survey on issues and enabling technologies. IEEE Internet of Things Journal. 2017;**4**(1):1-20. DOI: 10.1109/JIOT.2016.2615180

[29] Mesmoudi Y et al. A Middleware based on service oriented architecture for heterogeneity issues within the Internet of Things (MSOAH-IoT). Journal of King Saud University—Computer and Information Sciences. 2020;**32**(10):1108-1116. DOI: 10.1016/j.jksuci.2018.11.011

[30] Hammoudeh M et al. A service oriented approach for sensing in the Internet of Things: Intelligent transportation systems and privacy use cases. IEEE Sensors Journal. 2020; **21**(14):15753-15761. DOI: 10.1109/JSEN.2020.2981558

[31] Taneja M et al. SmartHerd management: A microservices-based fog computing-assisted IoT platform towards data-driven smart dairy farming. Software Practice and Experience. 2019;**49**:1055-1078. DOI: 10.1002/spe.2704

[32] Herrera-Quintero LF et al. Smart ITS sensor for the transportation planning using the IoT and Bigdata approaches to produce ITS cloud services. In: Proceedings of the IEEE 8th Euro American Conference on Telematics and Information Systems (EATIS); 28-29 April 2016; Cartagena, Colombia. New York: IEEE; 2016. pp. 1-7. DOI: 10.1109/EATIS.2016.7520096

[33] Kanti Datta S et al. IoT and microservices based testbed for connected car services. In: Proceedings of the IEEE 19th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM); 12-15 June 2018; Chania, Greece, Piscataway, NJ: IEEE; 2018. pp. 14-19. DOI: 10.1109/WoWMoM.2018.8449768

[34] Banerjee A, Jiang B. A Blockchain-based IoT platform integrated with cloud services. In: Proceedings of the International Conference on Parallel and Distributed Processing Techniques & Applications; July 29th - August 1st 2019; Las Vegas, Nevada. C. S. R. E. A., 2020.

[35] Liu T, Martonosi M. Impala: A middleware system for managing autonomic, parallel sensor systems. ACM SIGPLAN Notices. 2003;**38**(10):107-118. DOI: 10.1145/966049.781516

[36] Kwon Y, Sundresh S, Mechitov K, Agha G. ActorNet: An actor platform for wireless sensor networks. In:

Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems; 8-12 May 2006; Hakodate, Japan. New York: ACM, 2006. DOI: 10.1145/1160633.1160871

[37] Fok CL, Gruia-Catalin Roman GC, Lu C. Agilla: A mobile agent middleware for self-adaptive wireless sensor networks. ACM Transactions on Autonomous and Adaptive Systems. 2009;**4**(3):1-26. DOI: 10.1145/1552297.1552299

[38] Vasile-Marian Scuturici VM, Surdu S, Yann G, Petit JM. UbiWare: Web-based dynamic data & service management platform for AmI. In: Proceedings of the Posters and Demo Track Conference; 3 December, 2012; Montreal Quebec Canada. New York: ACM; 2012. DOI: 10.1145/2405153.2405164

[39] Aiello F, Fortino G, Galzarano S, Vittorioso A. TinyMAPS: A lightweight java-based mobile agent system for wireless sensor networks. In: Proceedings of the 5th International Symposium on Intelligent Distributed Computing (IDC 2011); October 2011; Delft, the Netherlands: Springer-Verlag Berlin Heidelberg; 2012. DOI: 10.1007/978-3-642-24013-3_16

[40] Kang P et al. Smart messages: A distributed computing platform for networks of embedded systems. The Computer Journal. 2004;**47**(4). DOI: 10.1093/comjnl/47.4.475

[41] Chekati A, Riahi M, Moussa F. Agent-based modelling approach for decision making in an IoT framework. In: Barolli L, Woungang I, Enokido T, editors. Advanced Information Networking and Applications. AINA; 2021. Lecture Notes in Networks and Systems, vol 226. Springer, Cham. DOI: 10.1007/978-3-030-75075-6_21

[42] Fortino G et al. An Agent-Based Middleware for Cooperating Smart Objects. In: Proceedings of the 11th International Conference on Practical Applications of Agents and Multi-Agent Systems; 22-24; May, 2013; Salamanca, Spain: Springer-Verlag Berlin Heidelberg; 2013. pp. 387-398. DOI: 10.1007/978-3-642-38061-7_36

[43] Rausch T, Nastic S, Dustdar S. EMMA: Distributed QoS-aware MQTT middleware for edge computing applications. In: Proceedings of the IEEE International Conference on Cloud Engineering (IC2E); 17-20 April, 2018; Orlando, FL, USA: IEEE; 2018. pp. 191-197. DOI: 10.1109/IC2E.2018.00043

[44] Pietzuch PR. Hermes: A scalable event-based middleware. University of Cambrige Computer Laboratory Technical Report N° 590; 2004. ISSN 1476-2986. Available from: https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-590.pdf

[45] Pramukantoro ES, Anwari H. An event-based middleware for syntactical interoperability in Internet of Things. International Journal of Electrical and Computer Engineering. 2018;**8**(5):3784. DOI: 10.11591/ijece.v8i5.pp3784-3792

[46] Sivaharan T, Blair G, Coulson G. Green: A configurable and reconfigurable publish-subscribe middleware for pervasive computing. In: Meersman R, Tari Z, editors. On the Move to Meaningful Internet Systems. Berlin: Springer; 2005. pp. 732-749. DOI: 10.1007/978-3-540-78731-0_4

[47] Costa P, et al. The runes middleware for networked embedded systems and its application in a disaster management scenario. In: Proceedings of the IEEE 5th Annual International Conference on Pervasive Computing and Communication (PerCom'07); 19-23 March 2007; White Plains, NY, USA: Computer Society; 2007; pp. 69-78. DOI: 10.1109/PERCOM.2007.36

[48] Meier R, Cahill V. Steam: Event-based middleware for wireless ad hoc networks. In: Proceedings of the IEEE

22nd International Conference on Distributed Computing Systems Workshops; 2-5 July 2002; Vienna, Austria: IEEE; 2002. pp. 639-644. DOI: 10.1109/ICDCSW.2002.1030841

[49] Lai S, Cao J, Zheng Y. Psware: A publish/subscribe middleware supporting composite event in wireless sensor network. In: Proceedings of the IEEE International Conference on Pervasive Computing and Communication (PerCom'09); 9-13 March 2009; Galveston, TX, USA: IEEE Computer Society; 2009. pp. 1-6. DOI: 10.1109/PERCOM.2009.4912862

[50] Silva JR, et al. PRISMA: A publish-subscribe and resource-oriented middleware for wireless sensor networks. In: Proceedings of the 10th Advanced IEEE International Conference on Telecommunications; 20-24 July 2014; Paris, France. International Academy, Research, and Industry Association (IARIA); 2014. pp. 87-97

[51] Boonma P, Suzuki J. TinyDDS: An interoperable and configurable publish/subscribe middleware for wireless sensor networks. In: Hinze A, Buchmann A, editors. Principles and Applications of Distributed Event-Based Systems. Hershey, Pennsylvania: IGI Global; 2010. p. 206. DOI: 10.4018/978-1-60566-697-6.ch009

[52] Levis P, Culler DE. Maté: A tiny virtual machine for sensor networks. In: Proceedings of the Tenth ACM International Conference on Architectural Support for Programming Languages and Operating Systems; 5-9 October 2002; San Jose, California, United States: ACM; 2002. DOI: 10.1145/605406.605407

[53] Koshy J, Pandey R. Vm: Synthesizing scalable runtime environments for sensor networks. In: Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems (SenSys '05); 2-4 November 2005; San Diego, California, USA: ACM; 2005. pp. 243-254. DOI: 10.1145/1098918.1098945

[54] Khalid Z, Fisal N, Rozaini M. A survey of middleware for sensor and network virtualization. Sensors. 2014;**14**(12):24046-24097. DOI: 10.3390/s141224046

[55] Costa N, Pereira A, Serodio C. Virtual machines applied to WSN's: The state-of-the-art and classification. In: Proceedings of the Second International Conference on Systems and Networks Communications (ICSNC 2007); 25-31 August 2007; Cap Eterel, France, IEEE Computer Society; 2007. pp. 50-50. DOI: 10.1109/ICSNC.2007.83

[56] Hong K et al. Tinyvm: An energy-efficient execution infrastructure for sensor networks. Software: Practice and Experience. 2012;**42**(10):1193-1209. DOI: 10.1002/spe.1123

[57] Mueller R, Alonso G, Kossmann D. SwissQM: Next generation data processing in sensor networks. In: Proceedings of the Third Biennial Conference on Innovative Data Systems Research (CIDR); 7-10 January 2007; Asilomar, CA, USA. Online Proceedings. Available from: www.cidrdb.org 2007. pp. 1-9. DOI: 10.3929/ethz-b-000004843

[58] Marques IL, Ronan J, Rosa NS. TinyReef: A register-based virtual machine for Wireless Sensor Networks. In: Proceedings of the IEEE International Conference on SENSORS; 25-28 October 2009; Christchurch, New Zealand: IEEE; 2009. pp. 1423-1426. DOI: 10.1109/ICSENS.2009.5398437

[59] de Freitas EP. A Survey on Adaptable Middleware for Wireless Sensor Networks. Halmstad University Technical Report IDE0851; 2008. Available from: http://www.diva-portal.org/smash/get/diva2:239429/FULLTEXT01.pdf

[60] Deshpande A, Suman N, Gibbons PB, Seshan S. IrisNet:

Internetscale Resource-Intensive Sensor Services. In: Proceedings of the ACM SIGMOD International Conference on Management of Data; 9-12 June 2003; San Diego, California, USA: ACM; 2003. p. 667. DOI: 10.1145/872757.872856

[61] Hasiotis T et al. Sensation: A middleware integration platform for pervasive applications in wireless sensor networks. In: Proceedings of the IEEE Second European Workshop on Wireless Sensor Networks; 31 January - 2 February 2005; Istanbul, Turkey: IEEE; 2005. pp. 366-377. DOI: 10.1109/EWSN.2005.1462028

[62] Madden S, Franklin MJ, Hellerstein JM, Hong W. TinyDB: An acquisitional query processing system for sensor networks. ACM Transactions on Database Systems. 2005;**30**(1):122-173. DOI: 10.1145/1061318.1061322

[63] Zhao D, Raicu I. HyCache: A user-level caching middleware for distributed file systems. In: Proceedings of the IEEE International Symposium on Parallel & Distributed Processing, Workshops and PhD Forum(IPDPSW); 20-24 May 2013; Cambridge, MA, USA: IEEE; 2013. pp. 1997-2006. DOI: 10.1109/IPDPSW.2013.83

[64] Han Q, Venkatasubramanian N. Autosec: An integrated middleware framework for dynamic service brokering. IEEE Distributed Systems Online. 2001;**2**(7):22-31

[65] Huebscher MC, McCann JA. Adaptive middleware for context aware applications in smart-homes. In: Proceedings of the 2nd Workshop on Middleware for Pervasive and Ad-hoc Computing; 18-22 October 2004; Toronto, Ontario, Canada, United States: ACM; 2004. pp. 111-116. DOI: 10.1145/1028509.1028511

[66] Alex H, Kumar M, Shirazi B. MidFusion: An adaptive middleware for information fusion in sensor network applications. Information Fusion. 2008;**9**(3):332-343. DOI: 10.1016/j.inffus.2005.05.007

[67] Grevenitis K et al. A hybrid framework for industrial data storage and exploitation. Procedia CIRP. 2019;**81**:892-897. DOI: 10.1016/j.procir.2019.03.221